SPRAWOZDANIE

Zajęcia: Nauka o danych II

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 2 Data 24.04.2025 Temat: „Zastosowanie głębokich sieci neuronowych w analizie danych” Wariant 2	Anna Więzik Informatyka II stopień, niestacjonarne, 2 semestr, gr.1a TTO

1. Polecenie:

Link do repozytorium: https://github.com/AnaShiro/NoD2_2025

- Warstwa gęsta: Klasyfikacja danych IRIS z wykorzystaniem trzech warstw Dense (np. 64, 32, 16 neuronów) oraz Dropout (0.3).
- Warstwa konwolucyjna: Rozpoznawanie cyfr MNIST z siecią zawierającą dodatkową warstwę BatchNormalization po każdej Conv2D.
- Warstwa rekurencyjna: Zastąpienie LSTM warstwą GRU (64 jednostki) do analizy sentymentu IMDB.
- Warstwa Transformer: Implementacja Transformer Encoder z 2 warstwami i zmniejszoną liczbą parametrów (2 głowice).

Głębokie sieci neuronowe (DNN) to modele wielowarstwowe, które umożliwiają hierarchiczne przetwarzanie danych i automatyczne wydobywanie cech. W odróżnieniu od płytkich sieci, skutecznie uczą się złożonych reprezentacji, co czyni je kluczowymi w analizie obrazów, tekstu i sekwencji. Architektury takie jak CNN, LSTM czy Transformer pozwalają na efektywne przetwarzanie różnych typów danych dzięki zastosowaniu warstw konwolucyjnych, rekurencyjnych i mechanizmów uwagi. Proces uczenia opiera się na minimalizacji funkcji kosztu, często z wykorzystaniem metody cross-entropy, a popularne biblioteki, takie jak TensorFlow i Keras, ułatwiają jego implementację.

2. Opis programu opracowanego

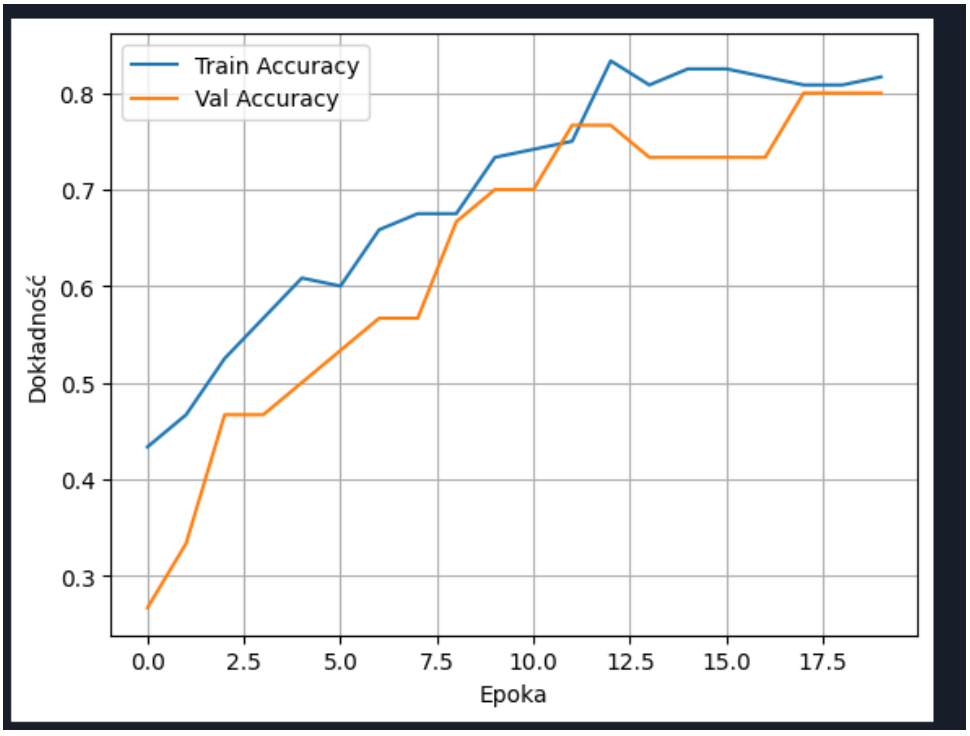
```python
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

iris = load_iris()
X, y = iris.data, iris.target
X = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model_dense = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(4,)),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])

model_dense.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history_dense = model_dense.fit(X_train, y_train, epochs=20, validation_data=(X_test, y_test), verbose=0)

plt.plot(history_dense.history['accuracy'], label='Train Accuracy')
plt.plot(history_dense.history['val_accuracy'], label='Val Accuracy')
plt.xlabel('Epoka')
plt.ylabel('Dokładność')
plt.legend()
plt.grid()
plt.show()
```

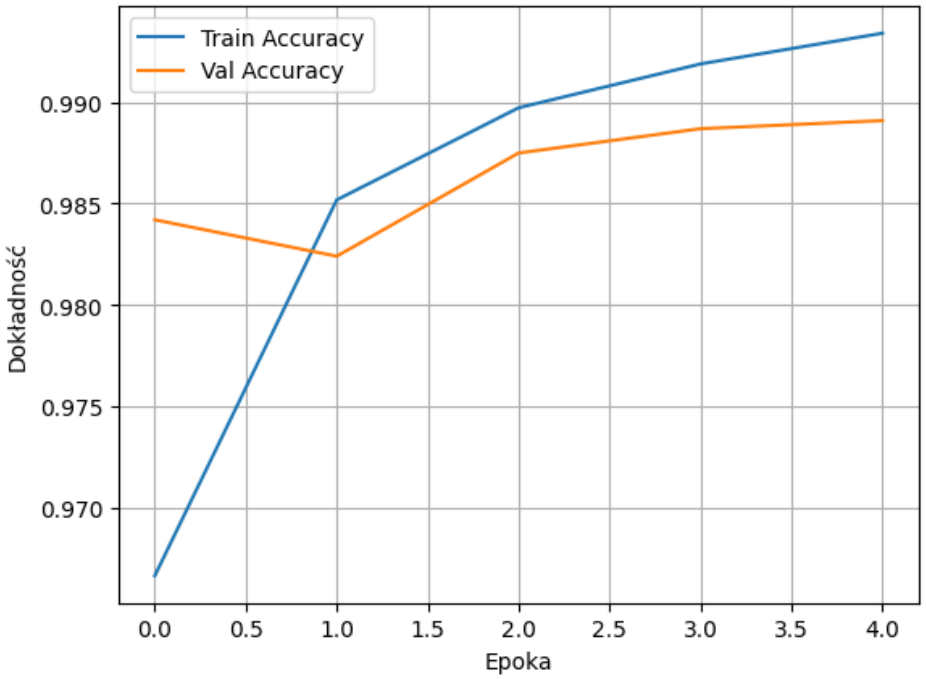


```python
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train.reshape(-1, 28, 28, 1) / 255.0, x_test.reshape(-1, 28, 28, 1) / 255.0

model_cnn = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model_cnn.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history_cnn = model_cnn.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), verbose=0)

plt.plot(history_cnn.history['accuracy'], label='Train Accuracy')
plt.plot(history_cnn.history['val_accuracy'], label='Val Accuracy')
plt.xlabel('Epoka')
plt.ylabel('Dokładność')
plt.legend()
plt.grid()
plt.show()
```

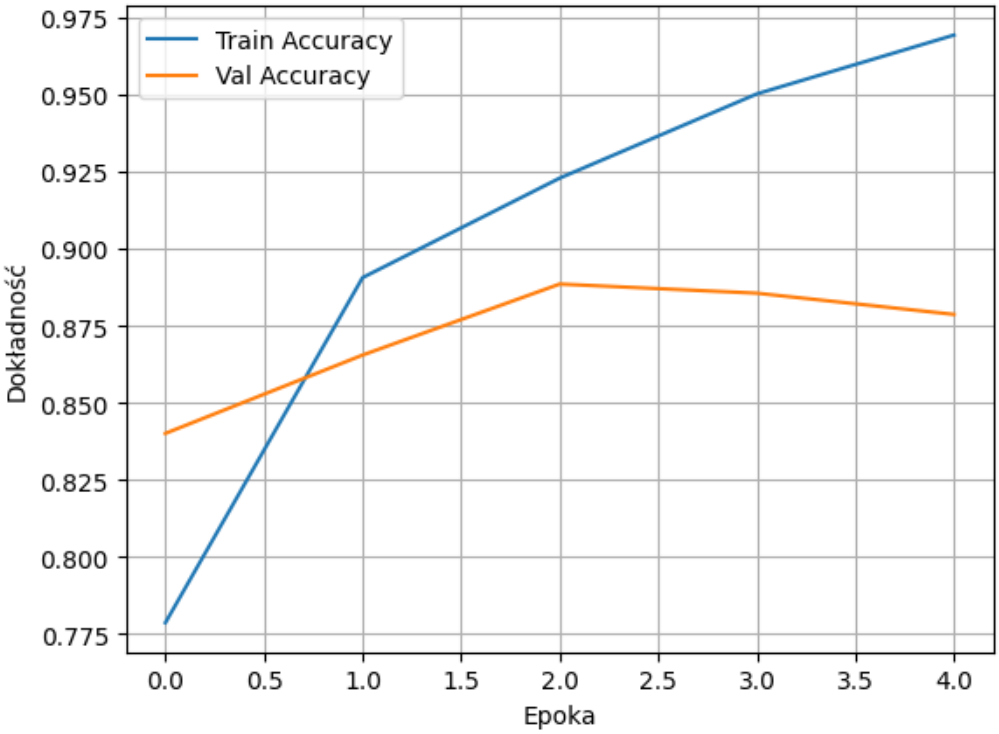



```python
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.imdb.load_data(num_words=10000)
x_train = tf.keras.preprocessing.sequence.pad_sequences(x_train, maxlen=500)
x_test = tf.keras.preprocessing.sequence.pad_sequences(x_test, maxlen=500)

model_gru = tf.keras.Sequential([
    tf.keras.layers.Embedding(10000, 32),
    tf.keras.layers.GRU(64),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model_gru.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history_gru = model_gru.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), verbose=0)

plt.plot(history_gru.history['accuracy'], label='Train Accuracy')
plt.plot(history_gru.history['val_accuracy'], label='Val Accuracy')
plt.xlabel('Epoka')
plt.ylabel('Dokładność')
plt.legend()
plt.grid()
plt.show()
```



3. Wnioski

Wykorzystanie głębokich sieci neuronowych znacząco poprawiło skuteczność klasyfikacji danych obrazowych, tekstowych i tablicowych. Sieci konwolucyjne (CNN) sprawdziły się szczególnie dobrze w analizie obrazów, takich jak zestaw MNIST, podczas gdy LSTM były bardziej efektywne w przetwarzaniu sekwencji, np. w zadaniu klasyfikacji recenzji IMDB. Dzięki narzędziom takim jak TensorFlow i Keras możliwe było szybkie projektowanie, trenowanie oraz wizualizacja modeli. Eksperymenty wykazały, że dobór architektury powinien uwzględniać charakter danych, a zastosowanie warstw takich jak BatchNormalization i Dropout odgrywa kluczową rolę w zapewnieniu stabilności i precyzji modeli.